CURSO PRÁTICO 51 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 35,00

INPUT

Vol. 4 Nº 51

NESTE NÚMERO

PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/ Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abilio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduíno F. Leite, Edson Donato

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 · 3º andar
CEP 01452 · São Paulo · SP · Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

AVALANCHE: CONTE OS PONTOS

- CONTAGEM DOS PONTOS
- TROCA DE TELA
- IMPRESSÃO DO ESCORE
- NÍVEL DE DIFICULDADE
- NÚMERO DE VIDAS

A simples obtenção de prêmios não satisfaz um aventureiro: ele precisa saber exatamente quantos pontos conseguiu com seu esforço. Monte estas rotinas e veja o placar funcionar.

A rotina que conta pontos também troca as telas, imprime o escore, o nível de dificuldade e o número de vidas.

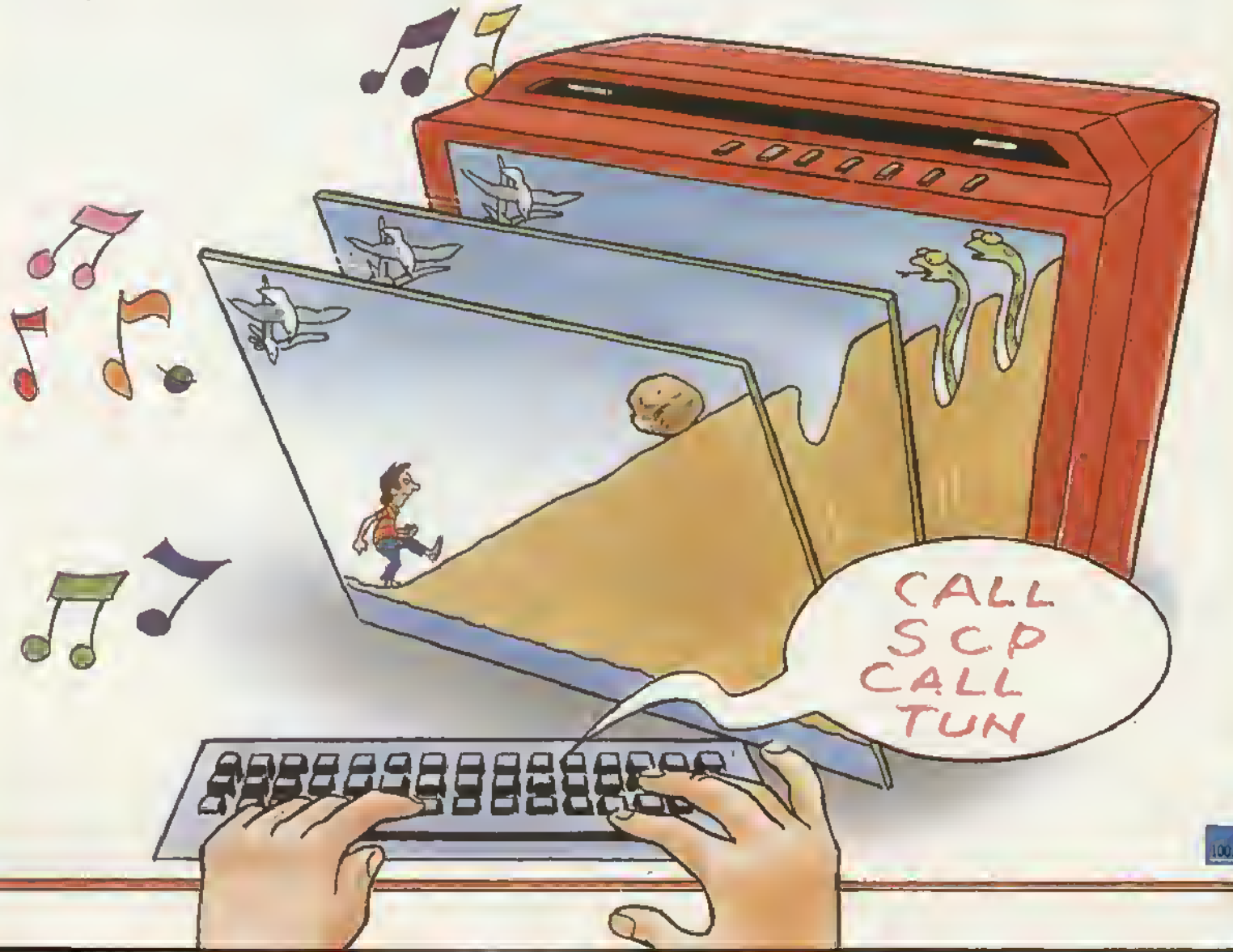
A rotina a seguir desenha a tela apropriada, imprimindo também o escore e o número de vidas que ainda restam a Willie. Além disso, encarrega-se da execução da música.

```
 10 REM org 58676
 20 REM call lsi
 30 REM call scp
 40 REM ld hl,119
 50 REM ld a,(57343)
 60 REM ld b,48
 70 REM add a,b
 80 REM call asc
 90 REM ld a,41
100 REM call print
110 REM call tune
120 REM ret
130 REM org 58303
140 REM lsi *
150 REM org 58174
160 REM asc *
170 REM org 58217
180 REM print *
190 REM org 60006
200 REM tune *
```

O INÍCIO DO JOGO

A instrução **call lsi** chama a sub-rotina que desloca a tela para a esquerda. A tela apropriada é selecionada pela sub-rotina **elb**, que publicamos em artigo anterior.

Em seguida, a rotina **scp**, que será dada logo adiante, é chamada. Ela é responsável pela impressão do escore na tela. Não execute o programa antes de tê-la montado.



O par de registros HL recebe a posição de impressão do número de vidas, 119. Quando a sub-rotina **print** for chamada, HL indicará, como de costume, a posição de impressão.

O acumulador recebe o conteúdo do endereço 57343. Nessa posição de memória armazena-se o número de vidas que Willie ainda tem. O número 48 é, então, colocado em B. Para calcular o código ASCII do número a imprimir, soma-se o conteúdo desses dois registros.

Depois a rotina **asc** é chamada. Como você deve estar lembrado, essa rotina faz com que BC aponte para o padrão do algarismo a ser impresso na tabela de caracteres da ROM. A cor do número é escolhida colocando-se 41 em A. A rotina **print** é chamada para imprimir na tela o número de vidas que restam a Willie. A rotina **tune**, por sua vez, é chamada para executar a música. Essa rotina deve ser montada novamente com endereço inicial 60000 — caso contrário, o programa não funcionará.

O PLACAR

Esta é a rotina **scp** chamada pela rotina anterior:

```
10 REM org 58939
20 REM scp ld hl,55
30 REM ld ix,57337
40 REM ld b,6
50 REM scq push bc
60 REM ld a,(ix+0)
70 REM ld b,48
80 REM add a,b
90 REM call asc
100 REM ld a,41
110 REM call print
120 REM inc hl
130 REM inc ix
140 REM pop bc
150 REM djnz scq
160 REM ret
170 REM org 58174
180 REM asc *
190 REM org 58217
200 REM print *
```

O par de registros HL recebe o valor 55, que corresponde à posição da tela em que será impresso o primeiro dígito do escore. O endereço do primeiro byte que contém o escore é colocado no registro-índice IX, que pode, então, ser usado como apontador.

O número de dígito do escore, 6, é colocado em B. Este contador fica temporariamente na pilha.

O byte apontado por IX é transferido em seguida para o acumulador, o que faz com que os dígitos do escore sejam colocados em A.

O número 48 é adicionado para calcular o código ASCII do algarismo e a rotina **asc** é chamada para acertar o apontador de padrões. Para a seleção da cor da letra, coloca-se 41 em A. A rotina **print** é chamada de novo para imprimir o dígito.

O par de registros HL aumenta em uma unidade para apontar a próxima posição de impressão. Observe que os dígitos mais significativos do escore são impressos primeiro. O registro IX também aumenta em uma unidade, apontando para a próxima variável do escore. Os dígitos decimais do placar são armazenados separadamente, um em cada variável, do mais significativo para o menos significativo.

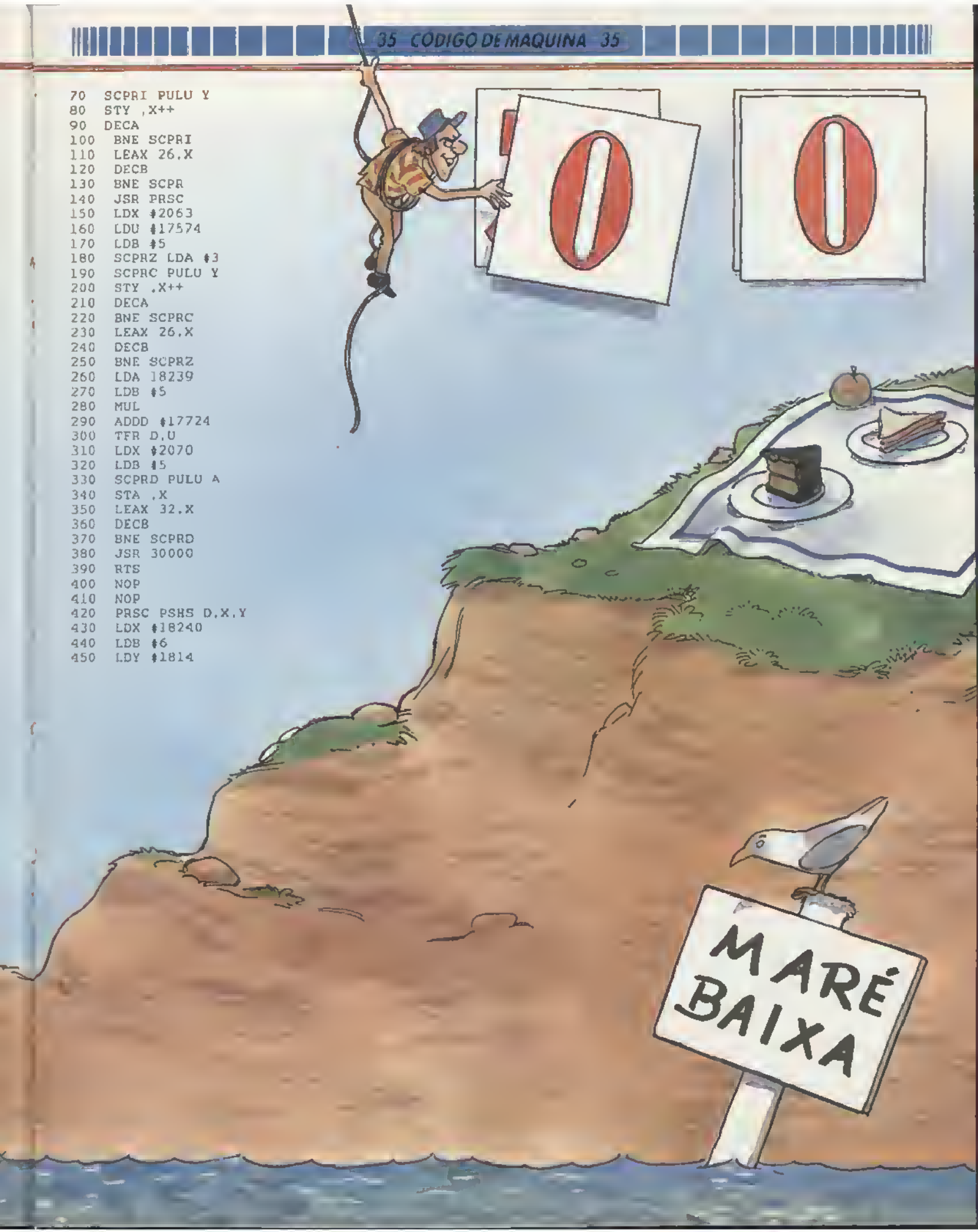
O contador é recuperado da pilha e colocado em B. A instrução **djnz** diminui seu valor em uma unidade. Se ele ainda não for zero, o processador volta ao rótulo **seq** para imprimir o dígito seguinte. Depois que o laço tiver sido executado seis vezes, imprimindo os seis dígitos do escore, o processador retornará.

T

Esta rotina desloca a tela antiga e desenha a nova com o sol, o número de vidas e o placar.

```
10   ORG 19489
20   JSR $4AA5
30   LDX #1807
40   LOU #17544
50   LOB #5
60   SCPR LDA #3
```

```
70  SCPRI PULU Y
80  STY ,X++
90  DECA
100  BNE SCPRI
110  LEAX 26,X
120  DECB
130  BNE SCPR
140  JSR PRSC
150  LDX #2063
160  LDU #17574
170  LDB #5
180  SCPRZ LDA #3
190  SCPRC PULU Y
200  STY ,X++
210  DECA
220  BNE SCPRC
230  LEAX 26,X
240  DECB
250  BNE SCPRZ
260  LDA 18239
270  LDB #5
280  MUL
290  ADDD #17724
300  TFR D,U
310  LDX #2070
320  LDB #5
330  SCPRD PULU A
340  STA ,X
350  LEAX 32,X
360  DECB
370  BNE SCPRD
380  JSR 30000
390  RTS
400  NOP
410  NOP
420  PRSC PSHS D,X,Y
430  LDX #18240
440  LDB #6
450  LDY #1814
```



```
460   PRSC8 LDA ,X
470   PSHS X,B
480   BITB #1
490   BNE ROLL
500   LDB #5
510   MUL
520   ADDD #17724
530   TFR D,X
540   PSHS Y
550   LDB #5
560   PRSCA LDA,X+
570   STA,Y
580   LEAY 32,Y
590   DECB
600   BNE PRSCA
610   PULS Y
620   LEAY 1,Y
630   ROLRET PULS B,X
640   LEAX 1,X
650   DECB
660   BNE PRSCB
670   PULS Y,X,D
680   RTS
690   ROLL LDB #5
700   MUL
710   ADDD #17724
720   TFR D,X
730   PSHS Y
740   LDB #5
750   RLLA LDA,X+
760   PSHS A
770   ANDA #15
780   LSLA
790   LSLA
800   LSLA
810   LSLA
820   ORA #15
830   STA 1,Y
840   PULS A
850   LSRA
860   LSRA
870   LSRA
880   LSRA
890   ORA #$50
900   STA ,Y
910   LEAY 32,Y
920   DECB
930   BNE RLLA
940   PULS Y
950   LEAY 2,Y
960   BRA ROLRET
```

A TELA

Inicialmente, o programa salta para a sub-rotina do endereço $4AA5, que substitui a tela antiga pela nova e desenha o sol.

Em seguida, a posição da tela em que

queremos imprimir a primeira letra da palavra "SCORE" é colocada em X. O apontador da pilha do usuário, U, recebe o endereço inicial da tabela de dados correspondentes à palavra a ser impressa. Esses dados foram colocados ali pelo programa BASIC criador de tabelas, já publicado. O registro B recebe o valor 5, pois as letras têm apenas cinco bytes de altura.

Existem cinco letras em "SCORE" mais um espaço, totalizando seis. Porém, A recebe o número 3, já que usaremos o registro Y para imprimir dois bytes de cada vez.

Os dois bytes que se encontram no topo da pilha do usuário são colocados no registro Y pela instrução **PULU Y**. A pilha do usuário é, no momento, a região da tabela de dados apontada por U. Esses dois bytes passam a ocupar as posições de memória de vídeo apontadas por X. Em seguida, o registro X tem seu conteúdo aumentado em duas unidades, apontando, então, para as duas posições seguintes.

A instrução **DECA** diminui o acumulador em uma unidade e **BNE SCPRI** repete o laço até que todos os seis bytes tenham sido impressos — o que coloca na tela apenas a primeira linha da palavra. Depois disso, a instrução **LEAX 26,X** soma 26 ao conteúdo de X, movendo-o da ponta direita da linha impressa até a posição de impressão da próxima linha da palavra. Esta fica seis bytes para a esquerda, na linha inferior. Lembre-se de que existem 32 bytes por linha e seis letras na palavra: 32 − 6 = 26.

O registro B é diminuído em uma unidade pela instrução **DECB**. O processador retorna ao rótulo **SCPR** até que todas as cinco linhas da palavra tenham sido impressas. Finalmente, a instrução **JSR PRSC** salta em direção à sub-rotina de impressão do escore.

AS CINCO VIDAS DE WILLIE

A rotina que imprime a palavra "VIDAS" é quase idêntica à que imprimiu "SCORE". Apresenta apenas duas diferenças: a posição de impressão na tela — apontada pelo registro X — e a porção da tabela de dados utilizada — apontada por U. Apesar da nova posição na tela e dos diferentes valores da tabela de dados, o processo de impressão é exatamente o mesmo.

Ao contrário da anterior, essa rotina não salta para a sub-rotina de impressão do número de vidas ao terminar, pois esta última se encontra logo a seguir na memória.

O número de vidas que restam a Willie fica armazenado em 18239. O conteúdo dessa posição é colocado no acumulador e o número 5, em B. Em seguida, os dois registros são multiplicados.

Para imprimir o número de vidas na tela, o programa tem que obter na tabela de dados o padrão do algarismo correspondente. Cada caractere requer cinco bytes para definir seu padrão. Assim, para acharmos o endereço inicial do padrão do algarismo desejado, precisamos percorrer a tabela de dados em múltiplos de cinco.

O resultado da operação **MUL** — que multiplica os conteúdos de **A** e **B** — é colocado em D. O endereço inicial da porção da tabela que nos interessa — 17724 — é somado a esse resultado. Obtém-se, assim, o endereço inicial do padrão do algarismo correspondente ao número de vidas.

O valor calculado é transferido para o apontador da pilha do usuário, U, de forma que essa região da tabela se transforma na pilha.

No registro X, coloca-se novamente a posição de impressão na tela; no registro B, o número de bytes necessário para escrever o algarismo, 5. O primeiro byte do padrão é retirado da pilha, colocado em A e impresso na posição de tela apontada por X.

Desta vez o registro X é acrescido de 32. Como apenas uma figura vai ser impressa, o apontador X precisa ser colocado uma posição de tela abaixo, de modo que passe a apontar para a próxima linha de pixels da figura.

B é decrementado e o processador sai do laço, executando a instrução seguinte se todos os cinco bytes já tiverem sido impressos na tela.

Para finalizar, o processador salta para a rotina 30000, que toca a música, criando o clima adequado para se iniciar a aventura. No retorno dessa sub-rotina, a instrução **RTS** devolve o controle ao BASIC.

RTS será apagada por uma outra instrução, quando o programa completo estiver montado. Ela é seguida por duas instruções **NOP** — Nenhuma **OP**eração —, que nada executam, servindo apenas para guardar o lugar dos dois últimos bytes da instrução **JSR**. Esta apagará **RTS** e acionará o laço principal que controla o programa completo.

OS NÚMEROS DO PLACAR

Os bits de cada linha de uma figura completam um byte. Se você imprimir cada um desses bytes diretamente, deixando-os muito próximos, eles terminarão por interferir uns nos outros, tornando-se, então, ilegíveis.

Para contornar essa dificuldade, a rotina que imprime os dígitos do escore controla as figuras — neste caso, números — meio byte, criando um espaço entre eles e fazendo com que os dígitos se tornem legíveis.

A rotina **PRSC** começa colocando os conteúdos dos registros D,X,Y na pilha da máquina, de modo a preservar seus valores. Mais tarde, precisaremos apenas do valor armazenado em Y. Mas nosso procedimento, aqui, vale como um lembrete: quando se programa em linguagem de máquina, convém empilhar o conteúdo de todos os registros que serão utilizados na sub-rotina. Sempre existe a possibilidade de que, posteriormente, seja necessário armazenar um importante parâmetro em um registro. Na dúvida, convém empilhá-lo.

A posição de memória 18240 armazena o byte correspondente ao primeiro dígito do escore. O valor decimal de cada um dos seis dígitos do escore — do mais significativo ao menos significativo — está guardado em seis posições de memória, de 18240 até 18245.

Como seis figuras serão impressas, carrega-se B com o número 6; Y é carregado com a posição de impressão.

O conteúdo da posição de memória apontada pelo registro X é colocado no acumulador A. Esse registro está apontado para a memória do escore. Em seguida, os valores de X e B são preservados na pilha.

A instrução **BITB #1** verifica o bit zero do registro B. Se B tem um valor par — e o bit zero, portanto, não é 1 —, a instrução **BNE ROLL** desvia o programa para a sub-rotina ROLL, que desloca todas as outras figuras meio espaço para a direita. Mas se o valor de B é ímpar — e, conseqüentemente, o bit zero é 1 — o processador continua com a próxima instrução.

IMPRESSÃO

Para imprimir a figura na tela, o procedimento é o mesmo utilizado no número de vidas. Multiplica-se o dígito requerido por cinco e adiciona-se o resultado ao endereço-base da tabela de dados do primeiro dígito.

Desta vez, porém, o apontador obtido é transferido para X, enquanto o apontador da posição de impressão em Y é colocado na pilha.

Carrega-se em A o byte da tabela de dados que está sendo apontado e incrementa-se o registro X. Esse byte é armazenado na posição de tela apontada por Y. O valor 32 é adicionado ao conteúdo de Y, fazendo com que este aponte para a linha de bits logo abaixo na figura. B é decrementado e o processador retorna para colocar o próximo byte abaixo, até que todos os cinco bytes que compõem a figura do dígito tenham sido impressos.

Depois disso, o apontador da posição de tela é recuperado da pilha e incrementado, passando a apontar para a próxima posição à direita.

O valor do contador em B e o apontador de memória do escore X são novamente recuperados da pilha. X é incrementado para o processamento da próxima figura à direita, enquanto B é decrementado, contando as figuras a serem impressas. Se todos os dígitos ainda não tiverem sido impressos, o processador imprime o próximo.

Neste caso, os registros Y, X e D são recuperados da pilha com os valores anteriores ao início desta rotina. Depois, o processador retorna para a posição que a chamou na rotina principal do programa.

DESLOCAMENTO DOS NÚMEROS

Se o valor do registro B for par e a rotina ROLL foi chamada, o processador localiza, na tabela de dados, o início da figura adequada ao dígito. Multiplica, então, o dígito por cinco e adiciona o resultado ao endereço-base, transferindo o valor obtido para X. Em seguida, coloca a posição de impressão na pilha e carrega o registro B com o contador de bytes.

A instrução **LDA,X+** carrega o acumulador com um dos bytes que formam a figura e incrementa o registro X para apontar o byte seguinte, que é guardado na pilha da máquina.

A instrução **ANDA #15** executa a operação lógica AND entre o conteúdo de A, uma linha de pontos da figura e o número 15, que em binário é 00001111. Essa operação equivale a colocar uma "máscara" no conteúdo de A, onde o *nybble* (grupo de quatro bits) mais significativo é apagado e o menos significativo, preservado.

Quatro instruções **LSLA** — do inglês **L**ogic **S**hift **L**eft on **A**cumulator (deslocamento lógico sobre o acumulador) — empurram o nybble menos significativo para a posição ocupada pelo nybble mais significativo, que é perdido. Essa operação é feita bit por bit.

A instrução **ORA #5** efetua a operação lógica OR entre o resultado anterior em A e o número 5, colocando amarelo — cor de fundo — no nybble menos sig-

nificativo. Após receber esse "tratamento", a linha de bits é impressa na posição apontada por Y+1. Exibe-se, assim, na tela, a metade direita do padrão de bits, duas posições à direita da figura atual, ficando reservado o espaço onde posteriormente será colocada a metade esquerda.

O padrão de bits completo dessa linha é mais uma vez recuperado da pilha da máquina e quatro instruções **LSRA** deslocam o nybble mais significativo para a direita. A operação OR com o número hexadecimal 50 ajusta a cor de fundo desse byte para amarelo.

O byte resultante das operações acima é impresso na posição apontada por Y, ou seja, a metade esquerda do padrão de bits dessa linha é colocada na posição reservada. Com isso, as duas metades se juntam, formando a figura completa. As metades vagas desses dois bytes foram ajustadas com a cor de fundo amarela, de modo que temos, entre cada dígito do escore, um espaço de meio byte.

A instrução **LEAY 32,Y** adiciona o valor 32 a Y, para processar a linha de bits logo abaixo da figura. B é decrementado e o processador permanece no laço até que todas as cinco linhas de bits tenham sido impressas. O processador sai, então, do laço e adiciona 2 ao apontador Y.

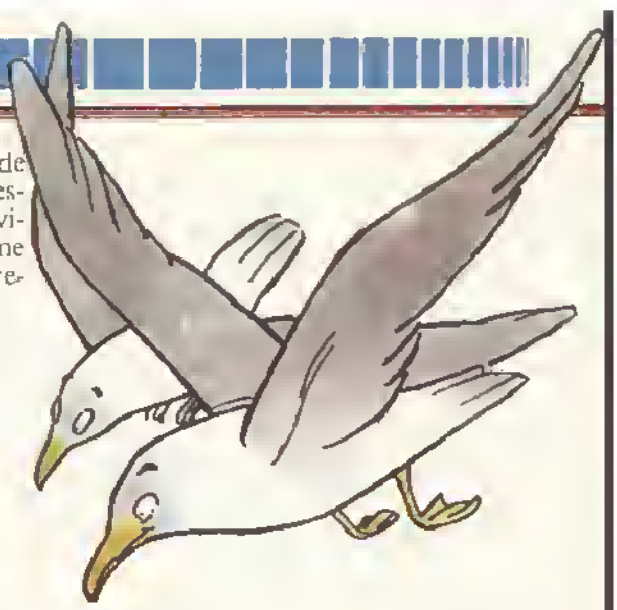
Isso faz com que o apontador de tela se desloque duas posições para a direita, já que a figura com número impar ocupou duas posições adjacentes na tela — uma contendo a metade esquerda e outra, a metade direita.

Em seguida, a instrução **BRA ROI.RET** retorna ao rótulo **ROI.RET**, no qual os apontadores são recuperados antes da rotina voltar para processar o próximo dígito, com número impar.

A rotina abaixo é montada logo após a que apresentamos no artigo anterior, e faz parte do programa principal. Ela começa chamando as rotinas -11973 e -11843. A primeira desenha e desloca a montanha para a tela, e a segunda acrescenta os prêmios e os riscos de acordo com o nível de dificuldade.

O programa pode ser dividido em três partes. A primeira cria os padrões de letras e números na VRAM, usando um *buffer* de códigos ASCII e uma rotina da ROM. A segunda imprime na primeira linha da tela as palavras SCORE, VIDAS e NÍVEL, utilizando a tabela de padrões. A terceira, finalmente, imprime os seis dígitos decimais do escore, segundo o conteúdo das seis posições de memória reservadas para eles; da mesma maneira, imprime o número de vidas e o nível atual do jogo, conforme o conteúdo das posições de memória reservadas para essas variáveis.

```
10    org 53961
20    call -11973
30    call -11843
40    ld a,160
50    ld (64695),a
60    ld a,16
70    ld (64697),a
80    ld de,-14560
90    ld b,52
100   rt push bc
110   push de
120   ld a,(de)
130   call 141
140   pop de
150   inc de
160   pop bc
170   djnz rt
180   ld de,1
190   ld a,84
200   ld b,30
210   tt push bc
220   push de
230   push af
240   ld hl,(62407)
250   add hl,de
260   call 77
270   pop af
280   inc a
290   pop de
300   inc de
310   pop bc
320   djnz tt
330   call sc
340   jp vd
350   sc ld de,-5219
360   ld (-5190),de
370   ld de,7
380   ld hl,(62407)
390   add hl,de
400   ld b,6
410   vo  push bc
420   push hl
430   ld de,(-5190)
440   ld a,(de)
450   inc de
460   ld (-5190),de
470   add a,126
480   call 77
490   pop hl
500   inc hl
510   pop bc
520   djnz vo
530   ret
540   vd ld a,(-5221)
550   add a,126
560   ld de,21
570   ld hl,(62407)
580   add hl,de
590   call 77
600   ld a,(-5228)
610   add a,126
620   ld de,30
630   ld hl,(62407)
640   add hl,de
650   call 77
```



```
660   call -12166
670   ld hl,-14054
680   call -12210
690   ret
700   end
```

ESCREVENDO NA TELA

Essa parte do programa começa na linha 40 e vai até a linha 320.

As posições de memória 64695 e 64696 da RAM contêm o endereço na VRAM do chamado *Acumulador Gráfico X*. A instrução **ld (64695),a** coloca o valor 160 nesse endereço. As posições de memória 64697 e 64698 contêm, por sua vez, o endereço do chamado *Acumulador Gráfico Y*. A instrução **ld (64697), a** coloca o número 16 nesse endereço.

A rotina 141 da ROM imprime um caractere ASCII em qualquer posição da tela de alta resolução, ou seja, do ponto (0,0) ao ponto (255,191). Para isso, o acumulador deve conter o código ASCII do caractere e os acumuladores gráficos x e y precisam estar ajustados. O valor do acumulador x vai de 0 a 255; o do acumulador y, de 0 a 191.

Como você deve se lembrar, a tela de alta resolução é um reflexo da tabela de nomes, que contém os códigos dos padrões que aparecem no vídeo. O conjunto de bytes que formam a figura de cada padrão está na tabela de padrões da VRAM. A rotina 141 da ROM escreve o caractere ASCII na tela. Para isso, precisa criar os bytes que formam a figura desse caractere na tabela de padrões e, ainda, colocar o código do padrão na posição adequada da tabela de nomes. A rotina 141 trabalha na tela como se o modo de alta resolução acabasse de ser ligado — ou seja, com a tabela de nomes (e, conseqüentemente, a tela) totalmente preenchida pelos padrões de 0 a 255, dispostos seqüencialmente. Escrever na tela de alta resolução com essa rotina significa, assim, criar os bytes

que compõem a figura no padrão ou padrões correspondentes à posição apontada pelos acumuladores gráficos x e y. Essa posição equivale ao canto superior esquerdo do caractere na tela. Por exemplo, com os valores 8 e 0 nos acumuladores gráficos x e y, respectivamente, a figura seria criada no padrão de código 1; com os valores 8 e 4, nos padrões 1 e 33. Como você pode concluir, um caractere ASCII ocupa até quatro padrões.

CRIAÇÃO DOS PADRÕES

Neste programa, utilizaremos a rotina 141 para criar os padrões das 52 letras e números cujos códigos ASCII estão a partir da posição de memória -14560. O programa BASIC que apresentamos no fim do artigo encarrega-se de colocar os caracteres nessas posições. Os bytes que compõem as figuras serão colocados a partir dos endereços da tabela de padrões que correspondem ao padrão 84. Para isso, deve-se carregar o acumulador gráfico x com 160 e o acumulador gráfico y, com 16.

O par de registros DE é usado como apontador do buffer de códigos ASCII e o registro B, como contador. Em seguida, o laço **rt** coloca as figuras correspondentes aos caracteres ASCII do padrão 84 ao padrão 135. Os acumuladores gráficos x e y não precisam ser incrementados, pois, a cada chamada, a rotina 141 cuida de fazer isso.

Se você transferir a posição de chamada das rotinas -11973 e -11843 para depois do laço **rt**, poderá ver na tela a criação dos padrões.

O próximo passo consiste na impressão das palavras SCORE, VIDAS e NIVEL na primeira linha da tela. Para isso, usa-se a rotina 77 da ROM, que coloca o código do padrão na tabela de nomes da VRAM. O código deve estar, então, no acumulador A, e o endereço na tabela de nomes, no par HL. As posições 62407 e 62408 contêm o endereço inicial da tabela de nomes da VRAM. Assim, basta colocar esse endereço em HL e adicionar a ele a posição na qual queremos imprimir o padrão, de 0 a 767.

A operação acima é repetida trinta vezes pelo laço **tl**, pois, como existem vários espaços entre as palavras que estamos escrevendo, elas ocupam quase toda a primeira linha.

ROTINA DE IMPRESSÃO

A rotina que imprime os dígitos do escore é chamada pela instrução **call sc**. Em seguida, o programa salta para a rotina que imprime o dígito do escore e o dígito do número de vidas. Procedemos assim para que a rotina sc possa ser chamada independentemente de outras partes do programa que serão dadas mais tarde.

A rotina **sc** começa colocando o endereço -5219 nas posições de memória -5190 e -5189, através do par DE. Os valores decimais dos dígitos do escore estão armazenados em seis endereços a partir de -5219.

EXIBIÇÃO DOS DÍGITOS

A posição de impressão do primeiro dígito é colocada no par de registros HL e o contador B é ajustado em seis (número de dígitos).

O laço **vo** coloca os dígitos na tela. As posições de memória -5190 e -5189 contêm o endereço do dígito que está sendo impresso e funcionam como apontador. Para obter o padrão equivalente ao dígito no acumulador, adiciona-se seu valor decimal ao código do padrão do dígito 0, que é 126, como se pode verificar contando os padrões criados anteriormente a partir de 84.

Os dígitos decimais correspondentes ao número de vidas e ao nível do jogo estão guardados nos endereços -5221 e -5228. Como os dígitos do escore, eles são colocados pela rotina 77 da ROM nas posições 21 e 30 da tabela de nomes, respectivamente.

Depois de tratar dos dígitos do escore, do número de vidas e do nível do jogo, a rotina chama a sub-rotina -12166 — a mesma que executa a música no início do jogo.

OS CÓDIGOS ASCII

O programa BASIC apresentado a seguir coloca os caracteres ASCII correspondentes às letras e números da linha **DATA** num buffer que começa no endereço -14560. Esse programa armazena todos os caracteres na variável **R$** e utiliza a função **ASC** para obter o código correspondente a cada um deles.

```
10 CLEAR 200,-16100
20 E=-14561
30 READ R$
40 FOR N=1 TO 52
50 POKE E+N,ASC(MID$(R$,N,1))
60 NEXT N
70 DATA "SCORE 000000  VIDAS 0
NIVEL  (FIM DE JOGO 0123456789'
80 END
```

TOCANDO EM HARMONIA

- GERAÇÃO DE ACORDES
- MELODIAS SIMULTÂNEAS
- A INSTRUÇÃO **SOUND**
- TABELA DE CONVERSÃO DE NOTAS PARA O MSX

Os usuários do MSX têm o privilégio de poder utilizar acordes em suas músicas, já que esse micro possui três canais para a produção de sons. Neste artigo, você verá como harmonizar suas melodias.

Em artigos anteriores, expusemos os fundamentos da teoria musical e mostramos como transformar o micro em um instrumento, por meio de programas simples. Vimos, também, como converter partituras em dados que possam ser lidos pela máquina. Tudo isso, porém, aplicava-se apenas a melodias em que uma só nota era emitida de cada vez.

Este artigo vai um pouco mais além, apresentando programas que possibilitam a execução de acordes. Embora a teoria envolvida seja de interesse geral, só o MSX tem os recursos necessários para a execução de mais de uma nota ao mesmo tempo, em BASIC.

Nossos programas adotam as mesmas convenções dos artigos anteriores. Assim, se você quiser refrescar a memória antes de tentar assimilar as novas técnicas, reveja-os.

O QUE É UM ACORDE?

Um acorde é simplesmente um grupo de notas tocadas simultaneamente. Se pressionarmos várias teclas ao acaso no piano, obteremos um acorde. O som resultante, contudo, poderá ser bem desagradável. Para que um acorde soe bem aos nossos ouvidos, é preciso haver *harmonia*. Os acordes mais simples e harmoniosos geralmente contêm três notas e se mantêm dentro de uma escala maior — dó, ré, mi...

Se tocarmos ao mesmo tempo dó, mi (duas notas acima) e sol (mais duas notas acima) — C, E e G na escala de C maior — teremos um exemplo do mais conhecido e simples tipo de acorde, o acorde maior. O nome de um acorde é derivado da nota mais baixa que o compõe: em nosso caso, trata-se de um acorde de C maior.

Todos os acordes maiores têm quatro semitons entre a nota mais baixa e a intermediária, e três semitons entre esta última e a nota mais alta. Um segundo tipo de acorde, o acorde menor, tem três semitons entre a nota mais baixa e a intermediária, e quatro entre as duas notas superiores. Tanto os acordes maiores como os menores possuem sete semitons de comprimento, só que a nota intermediária é mais alta no acorde maior. A conseqüência, em termos de qualidade sonora, é que os maiores são geralmente mais "alegres", e os menores, mais "tristes".

Podemos construir um terceiro tipo de acorde na escala maior. Começando em si, ou B em C maior, esse acorde conterá B, mais D e F da oitava superior. Ele terá, então, dois intervalos de três semitons entre as três notas. Este é um acorde diminuto, menos usado que os dois anteriores.

A escala maior compõe-se de sete notas. Com elas podemos construir três acordes maiores, três menores e um diminuto. Os três tipos são conhecidos como tríades, já que têm três notas cada. A nota mais baixa de cada tríade é chamada *fundamental*; a nota fundamental de E menor, por exemplo, é E.

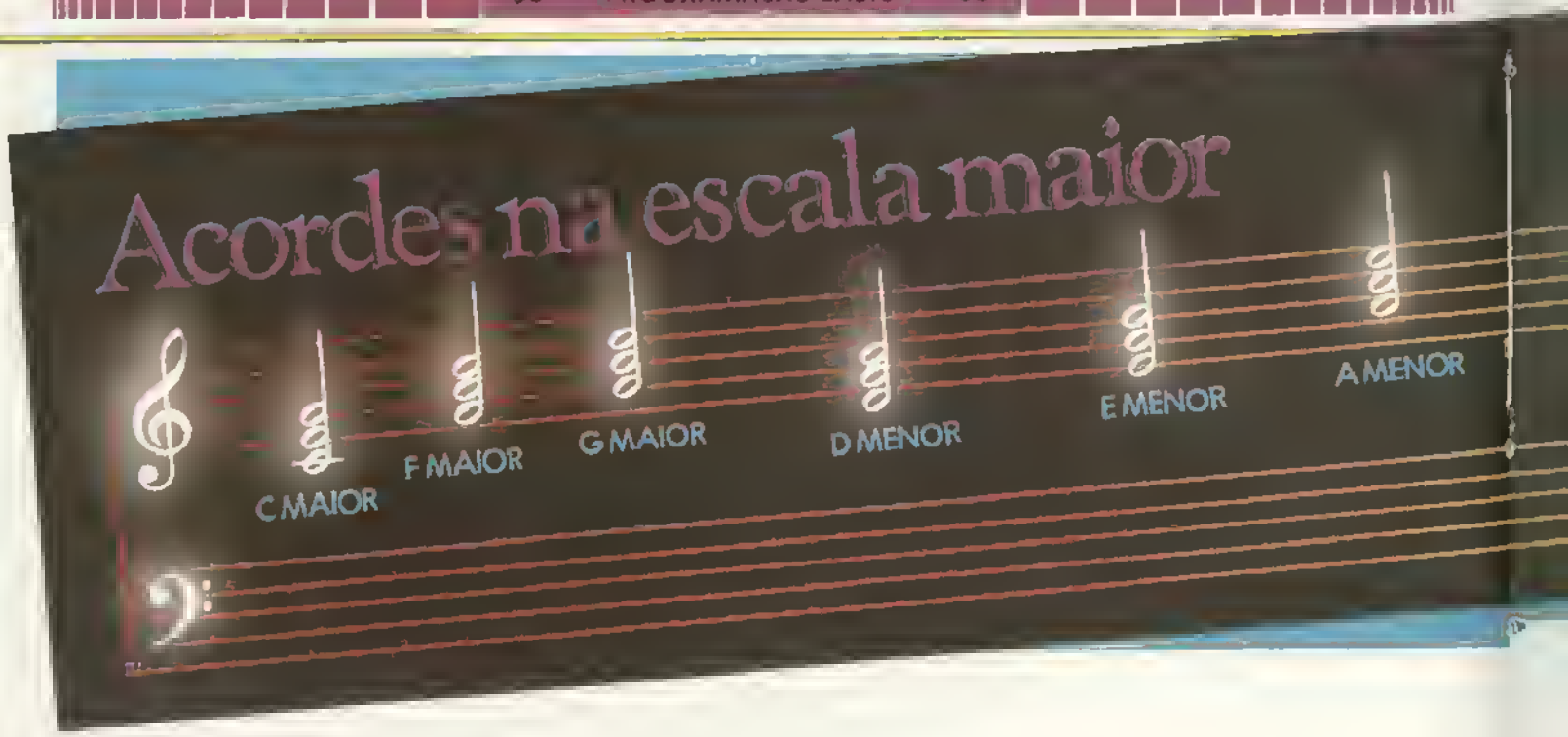
O diagrama acima mostra como esses acordes são escritos na pauta musical. Como você pode observar, as notas tocadas simultaneamente são colocadas umas sobre as outras.

COMO UTILIZAR OS ACORDES

Se uma melodia executada em C maior tem, em certo ponto, a nota C, qualquer acorde — dos tipos descritos — que também contenha a nota C poderá se harmonizar com ela, produzindo um som mais rico e agradável.

Como os acordes C e F maiores e A menor contêm a nota C, todos se harmonizarão com a melodia. A nota da música em questão pode ser qualquer uma das três notas que compõem o acorde — ou seja, toda nota se harmoniza com três diferentes tríades. C, por exemplo, é a nota mais baixa em C maior; é intermediária em A menor e é

Acordes na escala maior



a mais alta em F maior. Na realidade, se uma melodia contém a nota C, basta adicionarmos as outras duas notas para obtermos a tríade. Nesse caso, a nota da melodia original funciona não apenas como parte da melodia, mas, também, como parte do acorde.

No acorde de C maior, C não precisa ser necessariamente a nota mais baixa, podendo ter qualquer nota acima ou abaixo. Nesse caso, costuma-se dizer que o acorde é invertido. A ilustração da página mostra diversos arranjos de notas — todos eles são acordes de C maior. O mesmo princípio se aplica a outros acordes.

HARMONIA AUTOMÁTICA

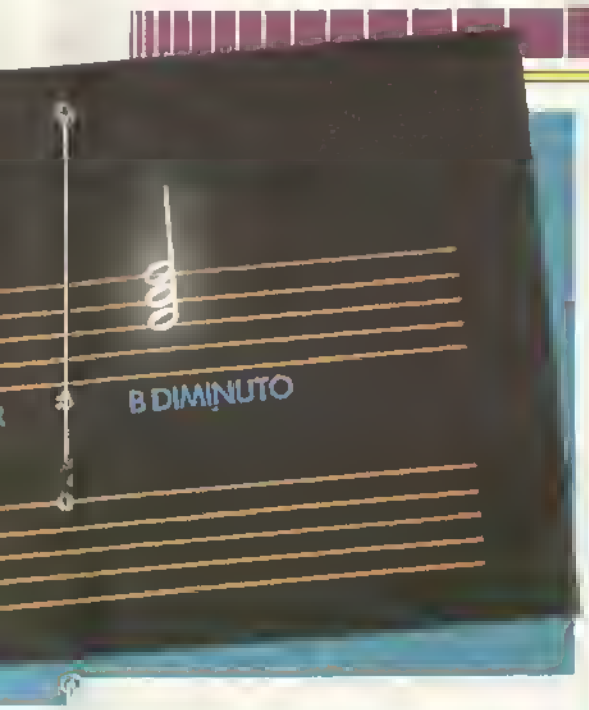
O programa que apresentamos a seguir obtém de linhas **DATA** as notas de uma melodia simples. Mas, ao executá-la, ele acrescenta a cada nota duas outras, mais graves, que harmonizam com ela. As duas notas são criadas por um processo aleatório; assim, a harmonia da música varia a cada nova execução. Todos os acordes, porém, constituem tríades da escala C maior.

A melodia listada ao final do programa poderá ser substituída por outra de sua própria escolha, desde que a escala seja C maior, isto é, o dó deve corresponder a O3C do comando **PLAY**. A nova melodia não deve conter nenhuma nota mais grave que O3C; caso contrário, o programa não será capaz de gerar um acorde válido.

As notas mudam ao mesmo tempo, nas três vozes e, sempre que há a alteração, uma nova harmonização é feita. O programa produz acordes de acordo com as regras, o que não significa que eles sejam sempre os mais adequados ao gosto do usuário.

A melodia *The Saints go Marching In* (*A Marcha dos Santos*) está armazenada nas linhas **DATA** do final do programa. A ilustração da página seguinte mostra duas variações da abertura da melodia que poderão eventualmente ser produzidas pelo programa. Trata-se da mesma melodia — o que muda é só a harmonia. Na primeira variação, os quatro acordes são F maior, C maior, B diminuto e G maior; na segunda, temos C maior, E menor, D menor e C maior.

```
10 R=RND(-TIME)
20 INPUT "Andamento (32-255)";T
P
30 GOSUB 3000
40 GOSUB 4000
50 GOSUB 5000
80 K1=1:K2=2:K3=3:K4=4:K5=5
90 FOR I=1 TO 32
100 READPV,T:IFPV=0THEN180
110 C=TB(PV)
120 I1%=RND(K1)*K2+K2:IFI1%=K3T
HENI2%=K5ELSEI2%=RND(K1)*K2+K4
130 I1%=C-I1%:I2%=C-I2%
140 A$="L"+STR$(T)+"N"+STR$(Q%(
PV))
150 B$="L"+STR$(T)+"N"+STR$(Q%(
TA(I1%)))
160 C$="L"+STR$(T)+"N"+STR$(Q%(
TA(I2%)))
170 PLAY A$,B$,C$
200 NEXT:END
4010 TM=23
4020 FOR I=1 TO 37
4030 Q%(I)=TM+I
4050 NEXT:RETURN
5000 DATA 1,3,5,6,8,10,12,13,15
,17,18,20,22,24,25,27,29,30,32,
34,36,37
5010 DIM TA(22):FOR I=1 TO 22:R
EAD TA(I):NEXT
5020 DATA 1,1,2,2,3,4,4,5,5,6,6
,7,8,8,9,9,10,11,11,12,12,13,13
,14
5030 DATA 15,15,16,16,17,18,18,
19,19,20,20,21,22
5040 DIM TB(37):FOR I=1 TO 37:R
EAD TB(I):NEXT:RETURN
3000 PLAY "V15","V10","V10"
3010 A$="T"+STR$(TP)
3020 PLAY A$,A$,A$
3140 RETURN
4000 DIM Q%(37)
10000 DATA 13,10,17,10,18,10
10010 DATA 20,2,13,10,17,10,18,
10
10020 DATA 20,2,13,10,17,10,18,
10
10030 DATA 20,5,17,5,13,5,17,5
10040 DATA 15,2,17,10,17,10,15,
10
10050 DATA 13,5,13,5,17,5,20,5
10060 DATA 20,10,18,2,17,10,18,
10
```




```
10070 DATA 20,5,17,5,13,5,15,5
10080 DATA 13,2
```

As linhas 10 a 80 iniciam o programa, encarregando-se de solicitar o andamento, chamar as sub-rotinas de inicialização e criar as variáveis K0 a K5, que terão valores de 0 a 5. Nas seções do programa onde a velocidade é muito importante (linhas 90-200), substituiremos os números por variáveis, já que as variáveis são manipuladas mais rapidamente que as constantes. Nessa parte do programa, todos os espaços entre os comandos devem ser removidos.

A sub-rotina da linha 3000 estabelece as características iniciais do comando **PLAY**. Note que cada canal é tratado separadamente.

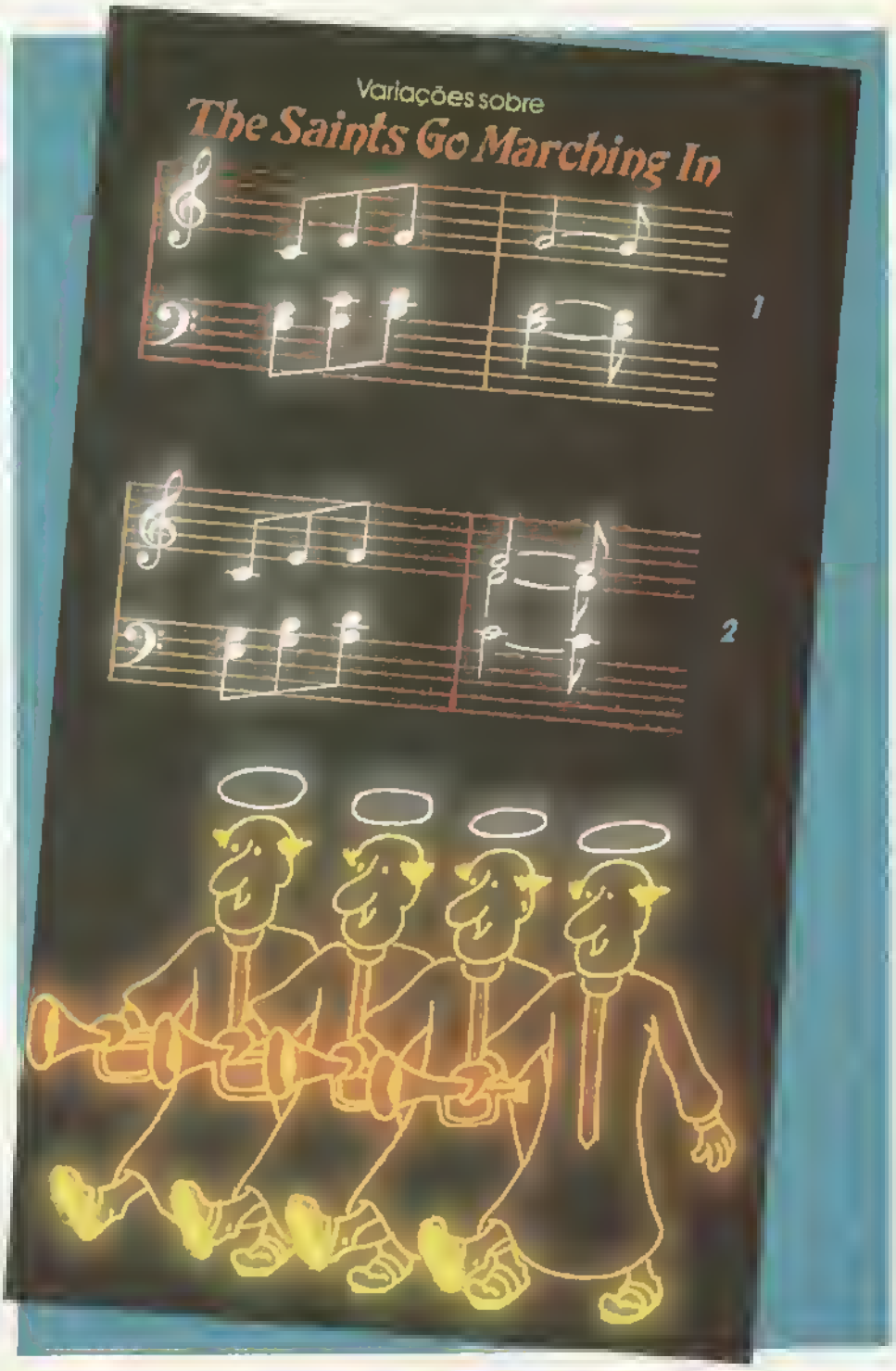
A sub-rotina da linha 4000 coloca na matriz **Q%** os números das notas. O comando **PLAY** pode utilizar número no lugar de notas, o que facilita bastante o cálculo da harmonia. Uma tabela de conversão de notas em números se encontra na página 1015.

A sub-rotina da linha 5000 cria duas matrizes, **TA** e **TB**, para definir os intervalos característicos da escala C maior, possibilitando, assim, o cálculo dos acordes. **TA** contém os números das notas da escala; **TB**, a posição da nota na escala. As duas matrizes permitem que o número da nota seja obtido a partir da posição da nota na escala e vice-versa, o que é importante para o cálculo automático da harmonia.

O laço principal do programa vai da linha 90 à 200. A linha 100 lê o valor da nota e sua duração. Se o valor for 0 — pausa —, o programa salta para a linha 180. A linha 110 calcula a posição da nota na escala usando a matriz **TB**. A 120 escolhe ao acaso um dos valores 2 a 3, definindo o intervalo entre as notas da melodia e a nota intermediária. Se o intervalo for 3, a nota mais baixa — **I2** — deve ficar cinco notas abaixo da nota da melodia. Se for 2, a nota mais baixa pode ficar quatro ou cinco notas abaixo, o que é decidido pela linha 130.

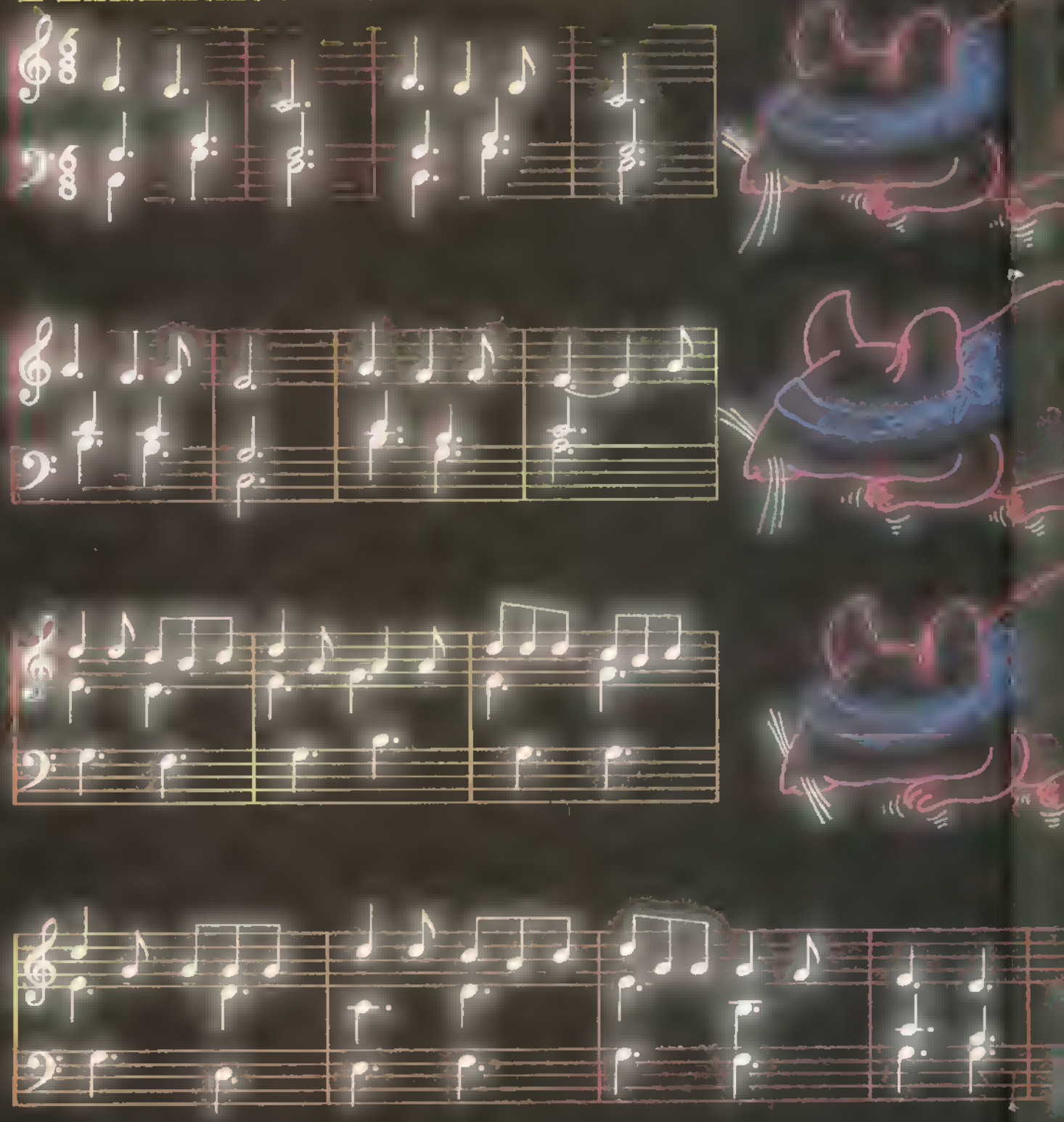
As linhas 140 a 160 definem os cordões que servirão de operando à instrução **PLAY** da linha 170. O valor da nota original é dado por **PV**, obtido das linhas **DATA**. O valor das duas notas restantes é calculado como auxílio da matriz **TA**, que converte a posição da nota na escala em seu valor real.

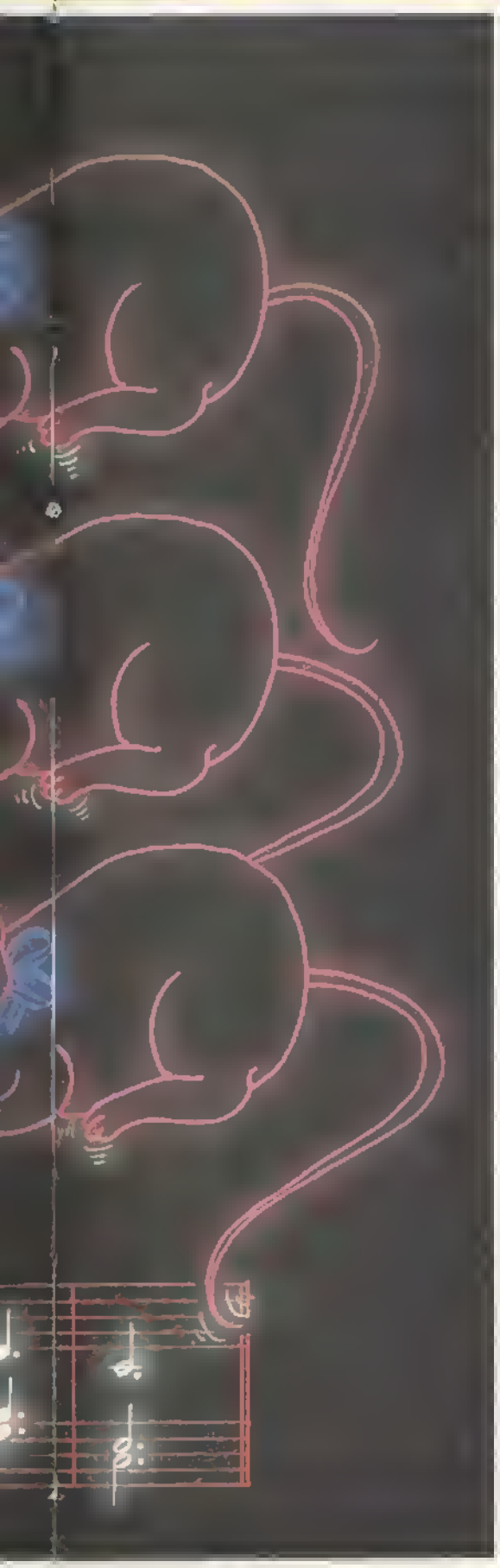
Observe que os números das notas



Arranjo para

THREE BLIND MICE





nas linhas **DATA** estão codificados conforme a escala geral. A matriz **Q%** é usada para convertê-los nos números efetivamente utilizados por **PLAY**. Os intervalos são operandos do mesmo comando — utilizados junto com a letra L — e não precisam ser convertidos.

MAIS ACORDES

O programa que apresentaremos agora especifica as notas tocadas por cada um dos canais. Como será possível qualquer combinação, deixaremos para o leitor os cuidados com a harmonia.

A música fica armazenada em linhas **DATA**, como de costume. Cada uma das linhas deve conter as notas do acorde juntamente com sua duração. Note que, muitas vezes, acordes consecutivos são muito parecidos, diferindo apenas pela nota de uma das vozes.

```
10 INPUT "ANDAMENTO (1-50)";TP
20 IF TP<1 THEN 10
30 A$="T"+STH$(INT(32+223/TP)):
PLAY A$,A$,A$
40 FOR I=1 TO 48
50 READ A$,B$,C$,D
60 A$="L"+STR$(INT(1+63/D))+"O3
"+A$
70 B$="L"+STR$(INT(1+63/D))+"O3
"+B$
80 C$="L"+STR$(INT(1+63/D))+"O3
"+C$
90 PLAY A$,B$,C$
100 NEXT
1000 DATA E,O2G,O2C,6
1010 DATA D,O2B,O2G,6
1020 DATA C,O2G,O2E,12
1030 DATA E,O2G,O2C,6
1040 DATA D,O2B,O2G,6
1050 DATA C,O2G,O2E,12
1060 DATA G,D,O2B,6
1070 DATA F,C,O2A,4
1080 DATA F,C,O2A,2
1090 DATA E,O2G,O2C,12
1100 DATA G,D,O2B,6
1110 DATA F,O2B,O2G,4
1120 DATA F,O2B,O2G,2
1130 DATA E,C,O2A,10
1140 DATA G,C,O2A,2
1150 DATA O4C,E,O2G,4
1160 DATA O4C,E,O2G,2
1170 DATA B,D,O2F,2
1180 DATA A,D,O2F,2
1190 DATA B,D,O2F,2
1200 DATA O4C,E,O2G,4
1210 DATA G,E,O2G,2
1220 DATA G,F,O2B,4
1230 DATA G,F,O2B,2
1240 DATA O4C,E,O2G,2
1250 DATA O4C,E,O2G,2
1260 DATA O4C,E,O2G,2
1270 DATA B,F,O2G,2
1280 DATA A,F,O2G,2
1290 DATA B,F,O2G,2
1300 DATA O4C,E,O2G,4
1310 DATA G,E,O2G,2
1320 DATA G,D,O2F,2
```

COMO MODIFICAR O TIMBRE

Além de produzir acordes, o MSX é capaz de modificar bastante as características do som obtido, fazendo-o ficar parecido com um instrumento específico, por exemplo.

Para isso, é necessário recorrer aos registros do PSG que determinam a curva envoltória do som. Trataremos detalhadamente desse assunto em um artigo futuro.

```
1330 DATA G,D,O2F,2
1340 DATA G,D,O2F,2
1350 DATA O4C,C,O2E,4
1360 DATA O4C,C,O2E,2
1370 DATA B,D,O2F,2
1380 DATA A,D,O2F,2
1390 DATA B,D,O2F,2
1400 DATA O4C,E,O2G,2
1410 DATA G,E,O2G,2
1420 DATA G,E,O2G,2
1430 DATA G,O2B,O2F,4
1440 DATA F,O2B,O2F,2
1450 DATA E,C,O2G,6
1460 DATA D,O2B,O2G,6
1470 DATA C,O2E,O2C,12
```

As linhas 10 e 20 encarregam-se do andamento. A linha 30 inicializa a função **PLAY**. A linha 50 lê as notas do acorde e sua duração nas linhas **DATA**. Esses dados são convertidos em operandos de instrução macromusical nas linhas 60 a 80. A linha 90 emite, efetivamente, o acorde.

Ao laço **FOR...NEXT** cabe evitar que uma mensagem de erro do tipo "OUT OF DATA" interrompa a melodia.

O COMANDO SOUND

Até aqui, utilizamos apenas a função **PLAY** para executar peças musicais. Contudo, o MSX dispõe de outro comando musical, muito mais versátil: **SOUND**. As características básicas desse comando foram explicadas no artigo da página 168. Sua função é alterar o conteúdo dos registros do gerador programável de som (PSG).

Para usarmos o comando **SOUND** na produção de sons musicais, precisamos saber os valores que devem ser colocados em seus registros para produzir as notas. Para isso, consulte a tabela de conversão que está na página 1015.

O programa a seguir toca acordes de

É possível adaptar os demais microcomputadores para a produção de acordes?

Os micros que não dispõem de um processador especial de som podem produzir, entre outros efeitos sonoros interessantes, acordes semelhantes aos obtidos nos computadores da linha MSX. Para isso, é necessário combinar a freqüência das diversas notas, o que requer o emprego de linguagem de máquina.

A técnica utilizada tem muita semelhança com as que foram explicadas nos artigos *Efeitos sonoros no Spectrum* (página 556) e *Apple e TK-2000: efeitos sonoros* (página 712). Embora não tenhamos abordado o assunto em INPUT, o mesmo pode ser aplicado ao micro TRS-Color.

duas notas. Os dez acordes iniciais da música *Greensleeves* estão listados nas linhas **DATA**. A melodia completa pode ser obtida no artigo da página 816.

```
10 FOR I=0 TO 10
20 READ A:SOUND I,A
30 NEXT
40 DATA 0,0,0,0,0,0,0,56,15,15
0
50 FOR I=1 TO 10
60 READ A,B,C,D,E
70 SOUND0,B:SOUND1,A
80 SOUND2,D:SOUND3,C
90 FOR J=1 TO 10*E:NEXTJ,I
100 PLAY "O5":END
110 DATA 0,253,0,253,10
120 DATA 0,213,0,253,30
130 DATA 0,189,0,213,10
140 DATA 0,169,0,213,15
150 DATA 0,159,0,213,5
160 DATA 0,169,1,28,10
170 DATA 0,189,1,28,30
180 DATA 0,225,0,225,10
190 DATA 1,28,0,225,15
200 DATA 0,253,0,225,5
```

MELODIAS SIMULTÂNEAS

O programa a seguir possibilita a execução de melodias simultâneas. Ele permite que os dados correspondentes a cada uma das vozes sejam armazenados em linhas **DATA** separadas. A melodia poderá, inclusive, ter uma velocidade diferente do acompanhamento.

Os dados para cada uma das três vozes devem terminar pelo par de valores 99,99. As outras linhas **DATA** contêm os valores das notas e suas respectivas durações. Cada linha **DATA** equivale a dois compassos. Para executar uma melodia com apenas duas vozes, basta colocar um segundo par 99,99 logo após o final dos dados da segunda voz.

O programa traz a melodia *Three Blind Mice* (*Os Três Ratinhos Cegos*) em um arranjo de três vozes mostrado no diagrama da página 1012.

```
10 R=RND(-TIME)
20 INPUT "Andamento (1-50)";TP
30 GOSUB 3000
40 GOSUB 4000
50 GOSUB 5000
80 K1=1:K2=2:K3=3
100 P1=0:P2=0:P3=0
110 GOSUB 1000
120 GOSUB 1100
130 GOSUB 1200
140 FOR I=1 TO 192
150 IF T1<>99THENT1=T1-K1:IFT1=
KOTHENP1=P1+K2:GOSUB1000
160 IF T2<>99THENT2=T2-K1:IFT2=
KOTHENP2=P2+K2:GOSUB1100
170 IF T3<>99THENT3=T3-K1:IFT3=
KOTHENP3=P3+K2:GOSUB1200
180 FOR DL=1 TO TP:NEXT
190 NEXT
200 PLAY "O5":END
1000 T1=DA%(K1,P1+K1)
1010 PV=DA%(K1,P1):IFPV=99ORPV=
KOTHENRETURN
1020 SOUND0,LQ%(PV):SOUND1,HQ%(
PV)
1030 RETURN
1100 T2=DA%(K2,P2+K1)
1110 PV=DA%(K2,P2):IFPV=99ORPV=
KOTHENRETURN
1120 SOUND2,LQ%(PV):SOUND3,HQ%(
PV)
1130 RETURN
1200 T3=DA%(K3,P3+K1)
1210 PV=DA%(K3,P3):IFPV=99ORPV=
KOTHENRETURN
1220 SOUND4,LQ%(PV):SOUND5,HQ%(
PV)
1230 RETURN
3000 FOR I=0 TO 10
3010 READ A:SOUND I,A
3020 NEXT
3030 DATA 0,0,0,0,0,0,0,56,15,1
0,10
3140 RETURN
4000 DIM HQ%(37),LQ%(37)
4010 TM=853:P2=2^(1/12)
4020 FOR I=1 TO 37
4030 LQ%(I)=TM-256*INT(TM/256):
HQ%(I)=TM/256
4040 TM=TM/P2
4050 NEXT:RETURN
5000 DIM DA%(3,1000)
5010 FOR VN=1 TO 3:P=0
5020 READ DA%(VN,P):READ DA%(VN
,P+1)
5030 P=P+2
5040 IF DA%(VN,P-2)=99 THEN NEX
T VN
5050 IF VN<4 THEN 5020
5060 RETURN
10000 DATA 17,6,15,6,13,12
10010 DATA 17,6,15,6,13,12
10020 DATA 20,6,18,4,18,2,17,12
10030 DATA 20,6,18,4,18,2,17,10
,20,2
10040 DATA 25,4,25,2,24,2,22,2,
24,2,25,4,20,2,20,4,20,2
10050 DATA 25,2,25,2,25,2,24,2,
22,2,24,2,25,4,20,2,20,2,20,2,2
0,2
10060 DATA 25,4,25,2,24,2,22,2,
24,2,25,2,20,2,20,2,20,4,18,2
10070 DATA 17,6,15,6,13,12
10080 DATA 99,99
20000 DATA 8,6,12,6,8,12
20010 DATA 8,6,12,6,8,12
20020 DATA 15,6,13,6,8,12
20030 DATA 15,6,12,6,13,12
20040 DATA 17,6,15,6,17,6,18,6
20050 DATA 17,6,18,6,17,6,15,6
20060 DATA 13,6,15,6,17,6,12,6
20070 DATA 13,6,12,6,5,12
20080 DATA 99,99
30000 DATA 1,6,8,6,5,12
30010 DATA 1,6,8,6,5,12
30020 DATA 12,6,10,6,1,12
30030 DATA 12,6,8,6,10,12
30040 DATA 8,6,6,6,8,6,12,6
30050 DATA 8,6,8,6,8,6,6,6
30060 DATA 5,6,6,6,8,6,6,6
30070 DATA 8,6,8,6,1,12
30080 DATA 99,99
```

As linhas 10 a 80 dão início ao programa. Encarregam-se de definir o andamento, chamar várias sub-rotinas e determinar as variáveis **K1**, **K2** e **K3**, que substituem os números 1, 2 e 3 onde a velocidade for crítica.

A sub-rotina 3000 acerta a configuração inicial do PSG, ativando-o para a produção de notas musicais nos três canais disponíveis. A primeira voz terá volume mais elevado que as demais.

A sub-rotina 4000 coloca nas matrizes **LQ%** e **HQ%** os valores dos bytes menos significativo e mais significativo, correspondentes às freqüências das notas musicais. Isso permite que as notas sejam definidas pelo seu número na escala geral, e não pelos complicados valores dos registros do PSG.

A sub-rotina 5000 lê os valores e durações das notas em linhas **DATA**, colocando-os na matriz **DA%** (3,1000). O primeiro índice corresponde ao número da voz, o segundo, à posição da nota na melodia. Esta será lida e depois executada, ao contrário dos programas apresentados anteriormente. **P1**, **P2** e **P3** funcionam como apontadores de cada uma das vozes.

A sub-rotina das linhas 1000 a 1030 toca uma nota no canal A; **P1** aponta o par de dados corrente que especifica nota e duração; **T1** é o contador que controla a duração da nota.

A linha 1010 coloca em **PV** o valor da nota. Se esse valor for 99, o programa chegou ao fim dos dados referentes ao canal A; se for zero, trata-se de uma pausa. Em ambos os casos, a sub-rotina termina numa instrução **RETURN**. Nas demais situações, a linha 1020 coloca os bytes calculados com o auxílio de **HQ%** e **LQ%** nos registros adequados do gerador programável de som, usando o comando **SOUND**. As sub-rotinas 1100 e 1200 executam as mesmas operações nos canais B e C.

A linha 100 acerta os ponteiros para começar a leitura de **DA%**. As linhas 110 a 130 chamam as sub-rotinas 1000, 1100 e 1200 pela primeira vez.

LAÇO PRINCIPAL

O **FOR...NEXT** entre a linha 140 e a linha 200 constitui o laço principal do programa. A linha 150 testa o contador **T1**, que contém o valor da duração da nota corrente no canal A: um valor 99 indica o fim dos dados. Se **T1** tiver outro valor, é diminuído em uma unidade. Quando chegar a zero, a execução da nota terá terminado e outra será obtida pela sub-rotina 1000. Enquanto o valor zero não for atingido, a nota continua soando.

As linhas 150 e 160 executam a mesma operação nas duas outras vozes. A linha 170 provoca um atraso proporcional ao andamento. Na linha 200, a instrução **PLAY "05"** é utilizada para interromper a música.

Observe que, quando utilizamos o comando **PLAY**, as durações eram definidas por durações de nota. Já com o **SOUND**, os valores são proporcionais à duração desejada para cada nota.

Para terminar, apresentamos as modificações que precisam ser feitas no programa que calcula a harmonia automática de acordes para que funcione com o comando **SOUND**:

```
20 INPUT "Andamento (1-50)";TP
140 SOUND0,LQ%(PV):SOUND1,HQ%(P
V)
150 SOUND2,LQ%(TA(I1%)):SOUND3,
HQ%(TA(I1%))
160 SOUND4,LQ%(TA(I2%)):SOUND5,
HQ%(TA(I2%))
170 FOR DL=1 TO (51-TP)*40/T:NE
XT
200 NEXT:PLAY "05":END
3000 FOR I=0 TO 10
3010 READ A:SOUND I,A
3020 NEXT
3030 DATA 0,0,0,0,0,0,0,56,15,1
0,10
4000 DIM HQ%(37),LQ%(37)
4010 TM=853:P2=2^(1/12)
4020 FOR I=1 TO 37
4030 LQ%(I)=TM-256*INT(TM/256):
HQ%(I)=TM/256
4040 TM=TM/P2
```

TABELA DE CONVERSÃO PARA O MSX

Escala geral	Byte mais significativo	Byte menos significativo	Número	Nota	Escala geral	Byte mais significativo	Byte menos significativo	Número	Nota
1	3	85	25	03C	20	1	28	44	G
2	3	37	26	C+	21	1	12	45	G+
3	2	247	27	D	22	0	253	46	A
4	2	205	28	D+	23	0	239	47	A+
5	2	165	29	E	24	0	225	48	B
6	2	127	30	F	25	0	213	49	05C
7	2	91	31	F+	26	0	201	50	C+
8	2	57	32	G	27	0	189	51	D
9	2	25	33	G+	28	0	179	52	D+
10	1	251	34	A	29	0	169	53	E
11	1	222	35	A+	30	0	159	54	F
12	1	195	36	B	31	0	150	55	F+
13	1	170	37	04C	32	0	142	56	G
14	1	146	38	C+	33	0	134	57	G+
15	1	123	39	D	34	0	126	58	A
16	1	102	40	D+	35	0	119	59	A+
17	1	82	41	E	36	0	112	60	B
18	1	63	42	F	37	0	106	61	06C
19	1	45	43	F+					

JOGOS DE GUERRA: PRIMEIROS PASSOS

A criação de jogos de guerra no computador é uma atividade simplesmente fascinante. Aprenda as técnicas envolvidas e depois mobilize seu micro para esta batalha de INPUT.

Embora fossem conhecidos há milhares de anos, os jogos de guerra (*Wargames*) estavam restritos, até há bem pouco tempo, aos quartéis-generais e escolas militares, devido à exigência de tabuleiros imensos e à grande quantidade de peças necessárias para representar as forças em conflito. O advento dos computadores permitiu que se popularizassem, já que a máquina confina toda a massa de dados em sua memória, utiliza a tela gráfica como mapa e pode até fazer o papel de jogador.

Esses jogos desenvolveram-se originalmente como elemento de apoio ao ensino de estratégia militar. Hoje, sua reprodução no computador tem seduzido uma multidão de usuários. E, embora o objetivo principal de um jogo de guerra seja sempre a eliminação do inimigo, ele se relaciona muito mais, quanto à diversão proporcionada, ao xadrez do que a um videogame de ação, cheio de tiros e pistolas laser.

Durante um jogo de guerra, a tela do micro exibe um mapa da região de batalha, indicando, em geral, as posições dos dois inimigos. O computador possibilita uma simulação da realidade muito mais fiel que a proporcionada pelos jogos tradicionais, na medida em que nos dá a chance de posicionar nossas forças sem que o inimigo as veja. As unidades só se tornam visíveis quando efetivamente descobertas.

Porém os jogos que os estrategistas utilizam costumam envolver os mínimos detalhes de uma guerra — o que não temos condições de fazer em um computador doméstico.

Alguns jogos de computador são destinados exclusivamente a adversários humanos. O programa que apresentamos em INPUT, porém, permite que você jogue contra seu micro.

PLANEJAMENTO

Uma guerra consiste em mover "unidades de combate" (em geral, soldados, mas, também, tanques ou canhões) até a posição do inimigo, para tentar submetê-lo ou destruí-lo.

Os ingredientes básicos de um jogo de guerra são as duas forças adversárias, seus movimentos e o combate entre as unidades. Com isso em mente, já podemos planejar o jogo. A batalha será na terra, no mar ou no ar? Queremos um jogo estratégico em larga escala, com muitos exércitos envolvidos; um jogo tático entre dois exércitos em um único campo de batalha; ou só escaramuças entre combatentes individuais?

O período da História em que se dá a guerra também é importante, pois determina o tipo de tecnologia bélica e as características dos combatentes. Podemos recriar batalhas famosas, como as travadas por Napoleão em território russo, ou inventar nossa própria guerra, dando asas à imaginação.

O passo seguinte consiste na definição das regras do jogo. Elas determinarão as circunstâncias que ajudarão ou prejudicarão cada um dos oponentes, estendendo-se aos menores detalhes incluídos no jogo. Você pode querer, por exemplo, que os soldados tenham a opção de usar armaduras. Isso os protegerá durante a luta, mas, em compensação, tornará mais lento o movimento de avanço ou de retirada.

É muito tentador enriquecer o jogo com mil detalhes, aproximando-o ao máximo da realidade. Porém o tamanho da memória limita a quantidade de detalhes e a complexidade das regras. Convém, assim, manter a atenção sobre os pontos principais:

- Local: informações, na forma de mapas, a respeito do local onde se encontram as tropas e do tipo de terreno da região — como afeta o movimento e que tipo de proteção oferece.

- As tropas: quantos homens são; que tipo de munição e proteção usam; a que velocidade se locomovem.

- Movimento: que distância uma unidade poderá se mover; como o tipo de terreno afetará o movimento.

- Comandos: definir como serão dadas as ordens e se as tropas poderão desobedecê-las ou não.

- O computador como inimigo: definir se a máquina será mais ou menos inteli-

- JOGOS DE GUERRA NO TABULEIRO E NO COMPUTADOR
- ORGANIZAÇÃO DO JOGO
- O CENÁRIO
- A ÉPOCA
- REGRAS DA BATALHA
- FUNCIONAMENTO DA TELA
- BLOCOS GRÁFICOS
- LIMPEZA DA ÁREA DE TEXTOS

gente, variando, assim, o nível de dificuldade do jogo.

• Combate: tipo de combate — balístico ou corpo-a-corpo; tipo de projétil — de simples flechas até mísseis intercontinentais.

NOSSO JOGO

Uma vez escolhidos o tipo e os componentes da guerra e o período histórico em que ela ocorre, precisamos determinar como tudo isso será representado no micro. Devemos considerar dois aspectos: o ponto de vista do jogador e o do computador, e como o jogo será apresentado a cada um deles.

Nesta série de cinco artigos tentaremos mostrar as técnicas básicas de programação criando em seu micro um jogo que denominaremos *Capa e Espada*. Ele consiste em uma batalha tática entre dois exércitos do período medieval. Essa guerra, contudo, não se passa em nenhuma data definida.

Como a maioria dos jogos de guerra, *Capa e Espada* mostra na tela um mapa, com a disposição das tropas e os acidentes geográficos da região. Os jogadores (no caso, você e o computador) agem como comandantes das unidades, devendo tomar as decisões estratégicas bem como dar as ordens relativas ao comportamento da tropa.

Instruções mais detalhadas serão fornecidas no quarto artigo desta série, quando o programa estiver completo. De um modo geral, cada jogador pode escolher entre dar ordens ou deixar as tropas como estão. O jogo se desenrola com os jogadores movimentando seus comandados pelo campo de batalha, alternadamente, a fim de organizar a disposição das forças. Esse movimento pode ou não resultar em conflito. O efeito dos eventuais choques entre as unidades é determinado pelo tipo e pelo poder das tropas envolvidas — além de uma pitadinha de sorte. O jogo continua até que um dos exércitos tenha sido praticamente destruído.

O programa mostrará as informações gráficas relevantes — o mapa com os exércitos — continuamente, reservando uma porção da tela para textos.

Usaremos blocos gráficos para criar o mapa, de maneira que instruções do tipo **PRINT** possam controlar tanto o mapa como a área de textos. No caso do TRS-Color, os blocos gráficos serão definidos por instruções **DRAW** na tela de alta resolução, como temos feito na maioria dos programas de INPUT. Utilizaremos também uma rotina para im-

primir textos na tela gráfica. No Apple e no TK-2000, instruções **POKE** controlarão o mapa, enquanto o texto ficará nas quatro linhas inferiores.

Neste jogo há quatro tipos de terreno: campo aberto, vilas, florestas e montanhas. Empregaremos espaços em branco para representar o campo — portanto, não será preciso um bloco gráfico especial. Usaremos dois tipos de bloco para representar as montanhas e um tipo para cada terreno restante.

Os exércitos terão oito unidades de combate: o líder e seus cavaleiros, os nobres vassalos (sargentos), duas unidades de lanceiros, duas de arqueiros e duas de camponeses. As duas primeiras unidades, formadas por nobres, requerem blocos gráficos diferentes. Para cada par de unidades restantes será usado um mesmo caractere duas vezes.

Teremos, assim, nove blocos gráficos: número um, vila; dois, florestas; três e quatro, montanha; cinco, o líder (representado por uma bandeira); seis, sargentos (representados por uma maça — arma medieval composta por um cabo com corrente e uma bola de ferro na ponta); sete, lanceiros (um escudo); oito, arqueiros (arco e flecha) e nove, camponeses (espada).

OS BLOCOS GRÁFICOS

Os programas listados a seguir criam os blocos gráficos descritos acima.

```
 210 FOR k=1 TO 9
 220 READ a$
 230 FOR I=0 TO 7
 240 READ a
 260 POKE USR a$+I,a
 270 NEXT I
 290 NEXT K
2540 REM Limpa tela de texto
2550 FOR k=17 TO 21: PRINT AT k
,0;"
     ": NEXT k
2570 DATA "a",16,16,60,126,255,
189,231,231
2590 DATA "b",16,56,84,16,56,84
,146,16
2610 DATA "c",8,20,34,65,6,8,16
,224
2630 DATA "d",0,48,72,132,2,0,0
,0
2650 DATA "e",128,240,255,252,1
43,128,128,128
2670 DATA "f",64,240,72,68,68,6
8,78,68
2690 DATA "g",255,231,231,129,1
29,231,102,60
2710 DATA "h",249,70,38,25,9,5,
3,1
2730 DATA "i",1,2,4,8,16,160,64
,160
```

```
200 SCREEN 1,2:KEY OFF:COLOR 1,
15,15
210 FOR I=0 TO 71
220 READ A
225 VPOKE BASE(7)+192*8+I,A
230 VPOKE BASE(7)+208*8+I,A
235 VPOKE BASE(7)+224*8+I,A
240 VPOKE BASE(7)+240*8+I,A
245 NEXT
250 FOR I=0 TO 1
260 VPOKE BASE(6)+24+I,4*16+15
265 VPOKE BASE(6)+26+I,6*16+15
270 VPOKE BASE(6)+28+I,12*16+15
280 NEXT
290 VPOKE BASE(6)+30,13*16+15
2570 DATA 16,16,60,126,255,189,
231,231
2590 DATA 16,56,84,16,56,84,146
,16
2610 DATA 8,20,34,65,6,8,16,224
2630 DATA 0,48,72,132,2,0,0,0
2650 DATA 128,240,255,252,143,1
28,128,128
2670 DATA 64,240,72,68,68,68,78
,68
2690 DATA 255,231,231,129,129,2
31,102,60
2710 DATA 249,70,38,25,9,5,3,1
2730 DATA 1,2,4,8,16,160,64,160
```

```
210 EE = 768:T = 16384: FOR I =
 EE TO EE + 30 * 8 - 1
220  READ A: POKE I,A
230  NEXT
2570  DATA  64,64,80,84,85,17,2
1,21
2575  DATA  0,0,2,10,42,34,42,4
2
2580  DATA  0,32,40,8,32,40,10,
2
2585  DATA  1,5,21,17,5,21,81,6
5
2590  DATA  64,48,12,3,0,0,112,
15
2595  DATA  1,6,24,96,120,7,0,0
2600  DATA  64,48,12,3,0,0,0,0
2605  DATA  1,6,24,96,0,0,0,0
2610  DATA  129,149,213,213,193
,129,129,129
2615  DATA  128,128,138,130,138
,128,128,128
2620  DATA  132,149,196,132,132
,132,196,132
2625  DATA  128,128,128,130,130
,130,138,130
2630  DATA  213,149,149,129,149
,148,208,192
2635  DATA  170,170,170,160,170
,138,130,128
2640  DATA 192,192,192,208,196,
193,213,128
2645  DATA 128,128,128,130,136,
160,170,128
2650  DATA  128,128,128,196,212
,148,149,145
2655  DATA  160,168,138,130,128
,128,128,128
2660  DATA  128,128,168,160,168
,128,128,128
2665  DATA  192,212,213,213,193
,192,192,192
2670  DATA  128,128,128,160,160
,160,168,160
2675  DATA  144,212,145,144,144
,144,145,144
2680  DATA  170,170,170,130,170
,168,160,128
2685  DATA  213,212,212,192,212
,148,133,129
2690  DATA  128,128,128,160,136
,130,170,128
2695  DATA  129,129,129,133,14
5,193,213,128
2700  DATA  130,138,168,160,128
,128,128,128
2705  DATA  128,128,128,145,149
,148,212,196
2710  DATA  0,0,0,0,0,0,0,0
2715  DATA  0,0,0,0,0,0,0,0
```

```
210 FOR K=1 TO 9
220 READ AS
230 UCS(K)=AS
290 NEXT K
2570 DATA BR2D2L2D5RU2R3D2R3U4L
2UL2D2
2590 DATA BR2R3DRDLDL2ND4L3U2R3
DL3
2610 DATA BD3E2R2UD2RFGLG3
2630 DATA BD3E2RF3
2650 DATA ND7FR2FD5UNR3L2
2670 DATA ND7FR4DNR2DNFL3UR2
2690 DATA R5ND6DL5D5R3DL
2710 DATA NR3DRF6U4LHE2DL
2730 DATA BR7G7E2UNF3H2DR
```

Todos os programas incluem várias linhas **DATA**, utilizando-as de uma forma diferente para a criação dos blocos gráficos. O Spectrum coloca seus blocos nos caracteres 124 a 133. O MSX usa a tela de texto de 32 colunas. Não recorreremos à tela gráfica porque não vamos empregar comandos de alta resolução, e é mais fácil escrever na tela de textos.

As linhas que vão de 210 a 245 desenham os blocos gráficos na tabela de padrões. Cabe às linhas 250 a 280 colorilos. São feitas quatro cópias dos nove blocos, uma de cada cor. Embora os blocos de terreno e os dois exércitos tenham cada qual uma só cor é mais fácil criar todos os blocos com as quatro cores do que definir as cores de blocos individuais.

O USO DO POKE

No Apple e no TK-2000, montamos os blocos gráficos com **POKE**, em vez de usar o comando **DRAW**. Fizemos isso para economizar memória, pois seriam necessários muitos bytes para definir uma tabela com tantas figuras. O banco de blocos ficará armazenado em uma região normalmente não utilizada da página três — assim, não ocupará espaço disponível ao BASIC. Na realidade, serão criados dezoito blocos, dois para cada unidade ou terreno, já que, para obter cor, só usamos colunas pares ou ímpares. Para maiores detalhes sobre esse tipo de bloco gráfico, veja os artigos das páginas 489 e 507.

A ÁREA DE TEXTOS

As linhas que se seguem são utilizadas para a impressão de mensagens no vídeo do micro.

```
2540 REM   Limpa tela de texto
2550 FOR k=17 TO 21: PRINT AT k
,0;"
     ": NEXT k
2555 RETURN
```

```
2540 REM Limpa área de texto
2550 FOR I=576 TO 767
2555 VPOKE BASE(5)+I,32
2560 NEXT:RETURN
```

```
2540  REM  LIMPA
2545  HOME : GOSUB 30000
2550  RETURN
```

```
2540 REM  LIMPA JANELA DE TEXTO
2550 PMODE 0,4:PCLS0:PMODE 3,1
2560 RETURN
```

O programa trata a tela como duas "janelas" — uma de textos, ocupando uma pequena área do vídeo, e outra gráfica, contendo o mapa da região e a disposição das tropas adversárias.

No desenvolvimento do jogo, a janela de textos precisará ser limpa e reescrita com freqüência. A janela gráfica, porém, é mais constante, exigindo apenas pequenas alterações nas posições das unidades.

Todos os microcomputadores, com exceção do Apple e do TK-2000, tratam a tela como uma unidade, de maneira que essas rotinas se encarregam de apagar exclusivamente a área de textos, deixando o mapa intacto.

No próximo artigo da série que aqui iniciamos veremos como desenhar o mapa da batalha e movimentar as unidades que se confrontam.

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Mexitronic I
Apple II +	Microcraft	Craft II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmer	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microangenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppe II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismec	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod. III	Digitus	DGT-100
TRS-80 Mod. III	Digitus	DGT-1000
TRS-80 Mod. III	Kemitron	Naja 800
TRS-80 Mod. III	Prologica	CP-300
TRS-80 Mod. III	Prologica	CP-500
TRS-80 Mod. III	Sysdale	Sysdale III
TRS-80 Mod. III	Sysdata	Sysdata Jr.
TRS-80 Mod. IV	Multix	MX-Compacto
TRS-80 Mod. IV	Sysdata	Sysdale IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Calt IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Appla Master	Brasil	Apple II +
Milmer	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymex	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sherp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microenganho I	Brasil	Apple II +
Spectrum	Microangenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdala	Sysdala III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81 — TRS-80 — TK-2000 — MSX

Spectrum — TRS-Color — Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NOVOS LANÇAMENTOS, NOVOS SUCESSOS.

JÁ NAS LIVRARIAS

A FRATERNIDADE DA PEDRA
David Morrell

Um grupo secreto, sob direção de um padre armado, passa a agir contra o terrorismo. Mas, será que violência se combate com mais violência? Eis o dilema de Drew, agente da Lei envolvido com fatos e figuras do mundo real, num livro surpreendente do criador de Rambo.

AVENTUREIROS E MILIONÁRIOS
Clark Howard

No romance do Texas, a saga da descoberta de um poço de petróleo e o drama de um casal que herda um lote de terra aparentemente sem valor e enfrenta com coragem os poderosos do lugar, até vencer. Uma vitória antes de tudo moral, numa história forte e envolvente, com realistas cenas de amor.

CAÇADA SEM FIM
Bryan Forbes

Uma brilhante história de espionagem envolvendo a KGB. Por que matar uma ex-espiã que já tinha sido desmascarada e torturada tempos atrás? Um agente inglês, seu antigo amante, enfrenta um desafio: descobrir por que ela foi morta... e por que *agora!*

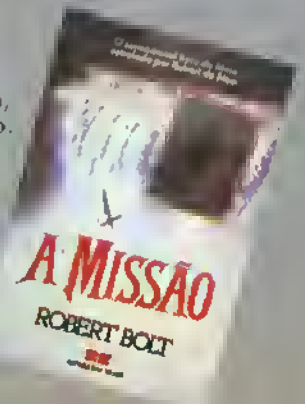
PODER
Howard Fast

Um líder sindical com a volúpia do poder, a luta pelos direitos dos trabalhadores, nos Estados Unidos, e sua manipulação por corruptos e oportunistas; o jogo das ambições políticas. Admiravelmente escrito, um romance atualíssimo.

SUPERSEXO
Alexandra Penney

Se não for o primeiro, este vai ser o último e definitivo guia para o prazer que o leitor poderá seguir: um livro que derruba mitos, faz sugestões provocantes e propõe técnicas ousadas para se chegar ao supersexo, uma relação intensa e especial entre os casais, que não exclui o romantismo.

Não perca também: A MISSÃO, de Robert Bolt, o livro do filme.



EDITORA BEST SELLER